土木建筑工程信息技术
Journal of Information Technology in Civil Engineering and Architecture

ChinaXiv合作期刊

基于 IFC 标准的 BIM 模型编程语言解析方法研究

陈 远 康 虹 张静雅

（郑州大学土木工程学院，郑州 450001）

【摘 要】IFC 标准定义了建筑信息模型交换的数据格式，为面向建筑全生命周期的信息交换和共享提供了标准的数据定义和数据模型，是目前对建筑信息描述最全面、最详细的标准，是解决建筑行业 BIM 软件之间数据交换性和互操作性问题的重要标准。但是 IFC 基于 EXPRESS 语言来定义建筑信息交换与共享，EXPRESS 语言本身不是编程语言且不可被计算机编译执行，因此使用计算机编程语言解析和处理基于 IFC 标准的 BIM 模型，是 BIM 软件开发的基础和关键技术。本研究利用开源的 Java 插件，解析基于 IFC 的 BIM 模型，生成相对应的 IFC 实体类，在此基础上，利用 Java 语言来具体编程实现不同的应用功能，为下一步的基于 IFC 标准的土木建筑工程 BIM 软件开发奠定了基础。

【关键词】建筑信息模型；BIM；IFC；Java
【中图分类号】TU17 【文献标识码】A 【文章编号】1674 - 7461(2017)03 - 0085 - 05
【DOI】10.16670/j.cnki.cn11 - 5823/tu.2017.03.15

chinaXiv:201712.01432v1

IFC(Industry Foundation Classes)标准是 buildingSMART International[1]制定和维护的建筑信息模型 BIM 数据交换标准，从 IFC4 开始，IFC 标准已经被 ISO 国际标准组织接受为 ISO 16739。IFC 是一个开放、标准化、支持扩展的通用数据模型标准，已被建设行业接受为国际标准，其目的是使 BIM 软件在建筑业中的应用具有更好的数据交换性和互操作性。IFC 是开放的建筑信息模型数据表达和交换标准，为面向建筑全生命周期的信息交换和共享提供了标准的数据定义和数据模型。IFC 基于 EXPRESS 语言来定义土木建筑工程领域信息交换与共享的数据表达。EXPRESS 语言是描述产品信息模型的标准语言，EXPRESS 不是编程语言并且不可以被计算机编译执行，但是可以被 JAVA、C + +、C# 等面向对象的编程语言识别和处理。因此，使用计算机编程语言识别和处理土木建筑工程领域基于 IFC 标准的数据表达和数据交换，以及 IFC 标准的计算机编程语言实现方法，是基于 IFC 标准的土木建筑工程 BIM 软件开发的基础和关键技术。

1 IFC 标准概述

开放性的国际标准 IFC 描述了建筑产品各方面的信息，定义了建筑信息交换的数据格式和建筑物及其附属物的信息交换格式，是目前对建筑信息描述最全面、最详细的标准。从 IFC 的 2X3 版本开始，加入了 GIS 数据信息以及 GUID，目前的最新版本是 IFC4。IFC 使用 EXPRESS 语言来描述建筑对象、实例关系、对象属性以及几何、度量等资源。IFC 采用 STEP Part21 文件描述 BIM 信息，即保存 BIM 信息的物理文件格式为 STEP Part21 格式。

当前被广泛支持的 IFC2X3 版本包含 600 多个实体定义，300 多个类型定义，IFC 标准的信息描述由四个层次组成，代表不同的级别，从底层到顶层包括：资源层、核心层、共享层和领域层，如图 1 所示。位于最底层的资源层用来描述基本属性的实体，包括几何信息资源、成本资源、拓扑资源、几何模型资源、结构荷载资源、几何约束资源等，通常作为上层实体的基础信息定义。核心层定义了模型

【作者简介】 陈远(1975 -)，男，博士，副教授，主要研究方向：土木建筑工程信息化。

ChinaXiv合作期刊

的核心框架，包括核心资源、控制扩展、产品扩展、过程扩展等。共享层包含最常用的实体以及跨专业交换的信息，包括共享的建筑构件组成部分、共享的管理组成部分、共享的设施组成部分等。领域层包括各领域具体的概念定义，包括建筑领域、结构构件领域、结构分析领域、电气领域等。

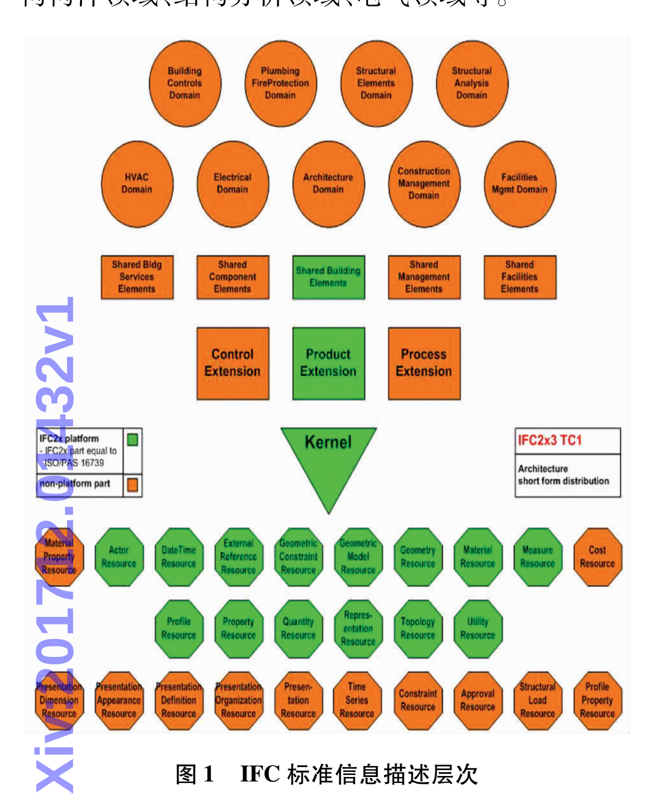


图 1　IFC 标准信息描述层次

chinaXiv:201712.01432v1

基于 IFC 标准的完整 BIM 模型由类型定义、函数、规则和属性集组成。IFC 模型的重要组成部分是类型定义，由实体类型、选择类型、枚举类型和定义类型四种类型组成。其中实体类型是 IFC 模型的核心，是信息交换和共享的载体，而其他类型是以属性值的方式作为实体类型属性的引用。在 IFC 层次体系中，核心层、共享层和领域层中的所有实体类型有一个共同的抽象基类 IfcRoot。IfcRoot 是处于顶层的抽象类，它有三个主要的继承类：IfcObjectDefinition、IfcPropertyDefinition 以及 IfcRelationship。这三个抽象类及其继承类构成了 IFC 模型的核心，处于核心层。而其他实体则主要由这三个抽象类继续派生，构成了不同领域和专业的实体类型，处于共享层和领域层。IfcObjectDefinition 的派生类 IfcObject 有七个主要的继承类，包括 IfcActor、IfcGroup、IfcProduct、IfcControl、IfcProcess、IfcResource 和 IfcProject。它们构成了 Ifc 模型的核心信息交换实体。IfcPropertyDefinition 及其继承类定义了 Ifc 模型常用的属性信息，并且提供了信息扩展方法。IfcRelationship 及其继承类为 Ifc 模型提供了实体与实体之间、实体与属性之间等各种复杂关系的定义。

2　IFC 模型的计算机编程语言解析方法

IFC 模型是基于 EXPRESS 语言描述的开放性的建筑信息数据格式。EXPRESS 语言是一种面向对象的、规范化的数据描述语言，重点在数据的描述和定义，EXPRESS 语言不是计算机编程语言，不能被计算机编译和执行。因此基于 EXPRESS 语言的 IFC 模型需要首先被解析，以生成相对应的 IFC 实体对象，建立实体对象之间的层次结构关系，并且通过 C#、C ++ 或者 Java 等计算机语言来具体编程实现不同的应用功能。目前基于 IFC 标准的 BIM 软件开发有很多，包括 IFC 模型浏览器[2]、基于 IFC 标准的 BIM 模型格式检查与验证[3]以及基于 IFC 标准的几何模型的解析[4]等。

有三种主要的解析方法可以将基于 EXPRESS 语言定义的 IFC 模型与具体的计算机编程语言进行绑定，以形成可供计算机调用的程序。这三种方法为早绑定、晚绑定和混合绑定。早绑定指在计算机编程语言中为 IFC 模型中的每一个实体创建具体的数据结构，并且能够通过编程访问具体的数据。晚绑定指通过 EXPRESS 数据字典来访问具体的数据，优点是实现简单，缺点为缺乏类型检查以及编程接口 API 不够友好。混合绑定指结合了两者的特点，但在具体软件开发中使用较少。

由于 IFC 是开放性的 BIM 标准，经过多年的发展，出现了许多商品化和开源的 IFC 开发工具和插件。例如典型的商业化工具平台 EPM Technology，包含的系列 IFC 开发工具有：EDMdeveloperSeat Basic，EDMdeveloperSeat Professional，EDMmodelMigrato，EDMmodelConverter 等。Eurostep 的 IFC Active Toolbox，以及 STEP Tools 的 ST – Developer 等。开源的 IFC 工具箱包括 BIMserver 和 IFC Tools Project 等。这些 IFC 工具及插件提供了较为完整的 EXPRESS 数据读写及访问功能，能够有效地处理 IFC 模型数据。通过使用这些工具和插件，使快速开发基于 IFC 标准的 BIM 软件成为可能，研究人员可以较少关注底层解析 IFC 模型的具体方法，而将大量的精力用在针对解决建筑行业具体问题的软件功能

ChinaXiv合作期刊

开发。

本文以其中 IFC Tools Project 为例，探讨基于 IFC 标准的 BIM 模型解析方法。IFC Tools Project[5] 提供了一个面向对象的早绑定 Java 插件，用来解析基于 IFC2x3 和 IFC4 的 BIM 模型。这个插件可以方便的读取、写入、更改和添加 IFC 文件，将 IFC 实体映射为 Java 类，并且对应的 Java 类包含了 IFC 实体的所有属性，以及支持 IFC 实体的反转属性。

这个插件主要由三部分组成：

(1) 每个 IFC 实体都有相对应的 Java 类。每个 Java 类都提供能够读取和设置对应 IFC 实体属性的方法，利用这些类的方法可以非常方便的读取和设置包括反转属性在内的所有属性。IFC 框架体系中所有实体的继承结构都体现到了相对应的 Java 类的继承结构。IFC 框架体系中所有定义类型和实体类型都映射为相应的接口。

(2) IFC 文件解析器，可以解析 IFC STEP 文件和 IFC Zip 文件。

(3) 核心 IFC 模型 IfcModel，用来读取 IFC 文件，操作 IFC 模型中所包含的 IFC 实体，增加与删除 IFC 实体。

通过调用 IfcModel 类的文件读取方法，IFC 模型将会被加载到内存并且所有 IFC 实体将会被初始化，如图 2 所示。

```
//创建IfcModel类的对象
IfcModel ifcModel = new IfcModel();
//从文件系统装载IFC文件
File stepFile = new File("d:\\revitpractice\\example.ifc");
try {
    ifcModel.readStepFile(stepFile);
} catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

图 2　IFC 文件读取与加载方法

以 IFC 标准为基础的 BIM 模型，其空间结构的组织形式包括项目、场地、建筑、楼层、空间等。其中 IfcProject 是整个模型的根实体，也是 IFC 模型解析的起点。图 3 展示了在 Java 程序中创建 IfcProject 类的实例，以及获得所有 IFC 实体对象的集合。每一个 IFC 实体的属性，都可以通过相对应的 Java 类包含的 get－和 set－方法读取和设置，如图图 4 所示，IfcWall 实体的 representation 属性可以通过对应的 IfcWall 类的 getRepresentation ( ) 方法获得。

```
// 创建IfcProject类的对象
IfcProject ifcProject = ifcModel.getIfcProject();
// 获得所有IFC实体对象的集合
Collection<ClassInterface> ifcObjects = ifcModel.getIfcObjects();
```

图 3　创建 IfcProject 类的实例，获得所有 IFC 实体对象的集合

```
IfcWall ifcWall = ... ;

//读取ifcWall的representation属性
IfcProductRepresentation representation = ifcWall.getRepresentation();
```

图 4　获得 IfcWall 实体的 representation 属性

3　IFC 模型解析实例分析

本文选取的 IFC 测试模型为 BuildingSMART 提供的 IFC 文件，版本为 IFC2x3，如图 5 所示。图 6 为对应的 BIM 模型。Java 开发环境为 Java 8、Eclipse Luna、SWT/JFace、RCP 等。IFC 测试文件包括 IfcWindow、IfcOpenElement 和 IfcWall 三个实体，每个实体都有自己的属性信息，包括宽度、高度、面积等。测试的目的为解析 IFC 文件，获取三个实体的物理属性信息，即 IfcPhysicalSimpleQuantity 实体所包含的数值信息。

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION (('ViewDefinition [CoordinationView,
QuantityTakeOffAddOnView]'), '2;1');
FILE_NAME ('example.ifc', '2015-01-07T15:18:06', ('Architect'), ('Building
Designer Office'), 'IFC Engine DLL version 1.02 beta', 'IFC Engine DLL version
1.02 beta', 'The authorising person');
FILE_SCHEMA (('IFC2X3'));
ENDSEC;
DATA;
#1 = IFCPROJECT('2BxD9ZTvnENhRLQgsovhgI', #2, 'Default Project', 'Description
of Default Project', $, $, $, (#20), #7);
#2 = IFCOWNERHISTORY(#3, #6, $, .ADDED., $, $, $, 1420615086);
#3 = IFCPERSONANDORGANIZATION(#4, #5, $);
#4 = IFCPERSON('ID001', 'Bonsma', 'Peter', $, $, $, $, $);
#5 = IFCORGANIZATION($, 'TNO', 'TNO Building Innovation', $, $);
#6 = IFCAPPLICATION(#5, '0.10', 'Test Application', 'TA 1001');
#7 = IFCUNITASSIGNMENT((#8, #9, #10, #11, #15, #16, #17, #18, #19));
#8 = IFCSIUNIT(*, .LENGTHUNIT., .MILLI., .METRE.);
#9 = IFCSIUNIT(*, .AREAUNIT., $, .SQUARE_METRE.);
#10 = IFCSIUNIT(*, .VOLUMEUNIT., $, .CUBIC_METRE.);
#11 = IFCCONVERSIONBASEDUNIT(#12, .PLANEANGLEUNIT., 'DEGREE', #13);
#12 = IFCDIMENSIONALEXPONENTS(0, 0, 0, 0, 0, 0, 0);
#13 = IFCMEASUREWITHUNIT(IFCPLANEANGLEMEASURE(1.745E-2), #14);
#14 = IFCSIUNIT(*, .PLANEANGLEUNIT., $, .RADIAN.);
#15 = IFCSIUNIT(*, .SOLIDANGLEUNIT., $, .STERADIAN.);
#16 = IFCSIUNIT(*, .MASSUNIT., $, .GRAM.);
#17 = IFCSIUNIT(*, .TIMEUNIT., $, .SECOND.);
#18 = IFCSIUNIT(*, .THERMODYNAMICTEMPERATUREUNIT., $, .DEGREE_CELSIUS.);
#19 = IFCSIUNIT(*, .LUMINOUSINTENSITYUNIT., $, .LUMEN.);
#20 = IFCGEOMETRICREPRESENTATIONCONTEXT($, 'Model', 3, 1.000E-5, #21, $);
#21 = IFCAXIS2PLACEMENT3D(#22, $, $);
#22 = IFCCARTESIANPOINT((0., 0., 0.));
#23 = IFCSITE('0g3TWscyX8beYSWxI_ZqQ0', #2, 'Default Site', 'Description of
Default Site', $, #24, $, $, .ELEMENT., (24, 28, 0), (54, 25, 0), $, $, $);
#24 = IFCLOCALPLACEMENT($, #25);
#25 = IFCAXIS2PLACEMENT3D(#26, #27, #28);
#26 = IFCCARTESIANPOINT((0., 0., 0.));
```

图 5　IFC 测试文件

chinaXiv:201712.01443v1

土木建筑工程信息技术
Journal of Information Technology in Civil Engineering and Architecture
ChinaXiv合作期刊

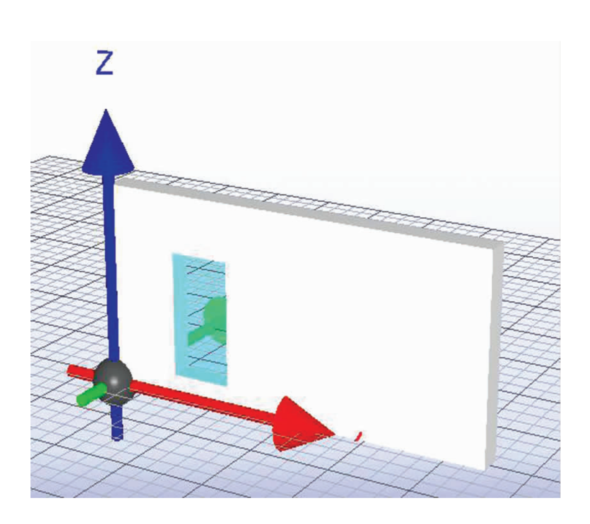


图 6　IFC 测试文件对应的 BIM 模型

如图 7 所示,在 IFC 文件的属性信息描述和关联机制中,属性集是通过 IfcDefinedByProperties 关系实体将 IfcObject 与 IfcPropertSetDefinition 描述的属性集信息相关联。IfcPropertSetDefinition 有三个派生类实体,其中 IfcElementQuantity 定义了建筑构件的物理属性信息,其通过 Quantities 属性,包含一个或多个 IfcPhysicalQuantity 实体。IfcPhysicalQuantity 有两个派生类,其中 IfcPhysicalSimpleQuantity 实体包含了单个的物理属性值,它有六个派生类实体,每个实体表达一种物理属性。例如 IfcQuantity-Length 代表构件的长度值,IfcQuantityArea 代表构件的面积等。

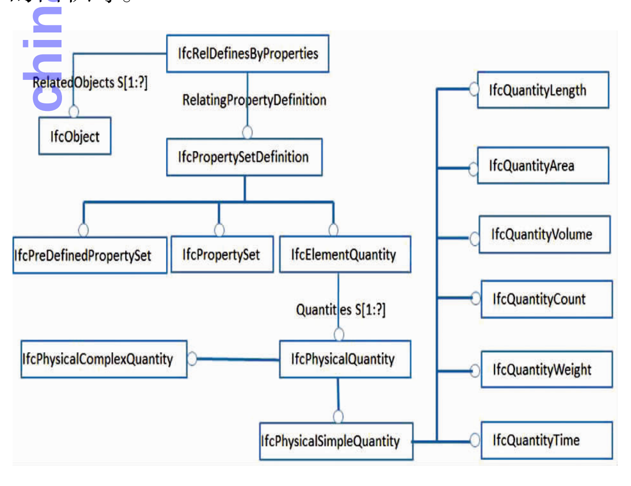


图 7　IFC 标准构件实体与属性集的关联机制

如图 8 所示,通过循环和迭代器,读取测试文件中的所有 IfcRelDefinesByProperties 实体,判断 IFC 标准的版本,以及进一步读取相关联的构件实体与属性集信息。

```
for (Iterator localIterator1 = this.ifcModel.getCollection(
      IfcRelDefinesByProperties.class).iterator(); localIterator1
      .hasNext();) {
    Object localObject;
    IfcElementQuantity localIfcElementQuantity;
    Iterator localIterator2;
    if ((localObject = (IfcRelDefinesByProperties) localIterator1
          .next()) instanceof IfcRelDefinesByProperties.Ifc2x3) {
      if (!(((com.ifctp.ifctoolbox.ifc.IfcRelDefinesByProperties.Ifc2x3) localObject)
            .getRelatingPropertyDefinition() instanceof IfcElementQuantity))
          continue;
      localIfcElementQuantity = (IfcElementQuantity) ((IfcRelDefinesByProperties.Ifc2x3) localObject)
            .getRelatingPropertyDefinition();
      for (localIterator2 = ((IfcRelDefinesByProperties.Ifc2x3) localObject)
            .getRelatedObjects().iterator(); localIterator2
            .hasNext();) {
          localObject = (IfcObject) localIterator2.next();

          a(localIfcElementQuantity,
                (IfcObjectDefinition) localObject);
      }
```

获得模型中所有的IfcRelDefinesByProperties实体
判断IFC版本
获得RelatingPropertyDefinition属性的值IfcElementQuantity
获得对应的构件实体

图 8　IFC 模型 IfcRelDefinesByProperties 实体解析方法

图 9 展示了通过循环和迭代器解析 IFC 模型 IfcElementQuantity 实体的方法。

```
if (paramIfcElementQuantity.getQuantities() != null) {
    for (Iterator localIterator = paramIfcElementQuantity
          .getQuantities().iterator(); localIterator.hasNext();) {
      if ((localIfcPhySicalQuantity = (IfcPhysicalQuantity) localIterator
            .next()) instanceof IfcPhysicalSimpleQuantity) {
          resolveSimpleQuantity(localIfcPhySicalQuantity,
                localHashMap);
      }
      if (!(localIfcPhySicalQuantity instanceof IfcPhysicalComplexQuantity))
          continue;
      resolveComplexQuantity(localIfcPhySicalQuantity, localHashMap);
    }
}
```

获得Quantities属性的值
判断Quantities属性值的类别

图 9　IFC 模型 IfcElementQuantity 实体解析方法

图 10 展示了通过判断和调用类相对应的方法,解析 IfcPhysicalSimpleQuantity 实体,以及读取建筑构件不同物理属性信息。

```
public void resolveSimpleQuantity(
      IfcPhysicalQuantity paramIfcPhysicalQuantity,
      HashMap<String, String> paramHashMap) {
   if ((paramIfcPhysicalQuantity = (IfcPhysicalQuantity) paramIfcPhysicalQuantity)
         instanceof IfcQuantityArea) {
      double d1 = ((IfcQuantityArea) paramIfcPhysicalQuantity)
            .getAreaValue().getValue();
      String str = this.decimalFormat.format(d1);
      paramHashMap.put(paramIfcPhysicalQuantity.getName()
            .getDecodedValue(), str);
      return;
   }
   if (paramIfcPhysicalQuantity instanceof IfcQuantityCount) {
      double d1 = ((IfcQuantityCount) paramIfcPhysicalQuantity)
            .getCountValue().getValue();
      String str = this.decimalFormat.format(d1);
      paramHashMap.put(paramIfcPhysicalQuantity.getName()
            .getDecodedValue(), str);
      return;
```

读取建筑构件的面积信息
读取建筑构件物理属性通过计算得来的数值信息

图 10　解析 IfcPhysicalSimpleQuantity 实体的方法
以及读取建筑构件不同物理属性信息

通过编程对 IFC 模型的结构体系进行深入解析,读取 IFC 实体的数据,并且将数据保存在 Java 容器中,最后用 Java JFace 中的 TableViewer 将所有建筑构件的物理属性信息显示出来,如图 11 所示。

chinaXiv:201712.01132v1

ChinaXiv合作期刊

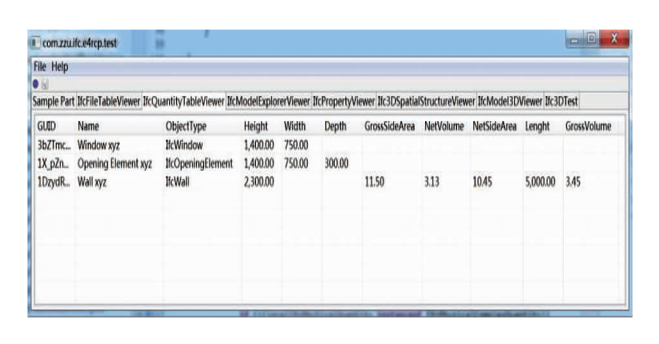


图 11　建筑构件的物理属性信息

4　结论

IFC 标准是基于 EXPRESS 语言描述的一种建筑信息模型数据表达格式。EXPRESS 语言不是计算机编程语言,不能被计算机编译和执行。本文利用开源的 Java 插件,首先解析基于 EXPRESS 语言的 IFC 模型,生成相对应的 IFC 实体类,建立实体类之间的层次结构关系,在此基础上,利用 Java 语言来具体编程实现不同的应用功能。研究人员可以较少关注解析底层 IFC 标准的具体方法,而将大量的精力用在针对解决建筑行业具体问题的软件功能开发,为下一步的基于 IFC 标准的土木建筑工程 BIM 软件开发奠定了基础。

chinaXiv:201712.01432v1

参考文献

[1] IndustryFoundation Classes IFC Official Release[R]. buildingSMART, http://www.buildingsmart－tech.org, 2015.

[2] Owolabi, A., Anumba, C. J., EI － Hamalawi, A., Harper, C., Development of an Industry Foundation Classes Assembly Viewer[J]. Journal of Computing in Civil Engineering. 2006, 20(2): 121-131.

[3] Zhang, C., Beetz, J., Weise, M., Model view checking: automated validation for IFCbuilding models[C]. eWork and eBusiness in Architecture, Engineering and Construction ECPPM. 2014:123-128.

[4] 张建平,张洋,张新. 基于 IFC 的 BIM 三维几何建模及模型转换[J]. 土木建筑工程信息技术,2009,1(1): 40-50.

[5] IFCTools Project, http://www.ifctoolsproject.com/, 2016.

Research on Resolving Method of IFC-based BIM Model Programing Language

Chen Yuan, Kang Hong, Zhang Jingya

(*School of Civil Engineering*, *Zhengzhou University*, *Zhengzhou* 450001, *China*)

Abstract: The IFC standard defines the data format for the exchange of BIM modelsand provides a standard data definition and data model for the information exchange and sharing of the whole building life-cycle. It is the most comprehensive and detailed standard for building information description and the key concern to solvethe data exchange and interoperability issues between BIM software. However, the IFC standard is based on EXPRESS language to define the building information exchange and sharing, and the EXPRESS language itself is not a programming language that fails to be compiled by the computer compiling. Therefore, it is the foundation and key technology of BIM software development to use the computer programming language to parse and process the IFC-based BIM model. This research uses the open source Java plug-in to analyse the IFC-based BIM model and generate the corresponding IFC entity class. Based on the resolving of IFC models, different software functions can be programmed by using Java language, based on which the next step of IFC-based BIM software development can be realised.

Key Words: BIM; IFC; Java